Using Guided Inspection to Validate UML Models

Melissa L. Major

Software Architects

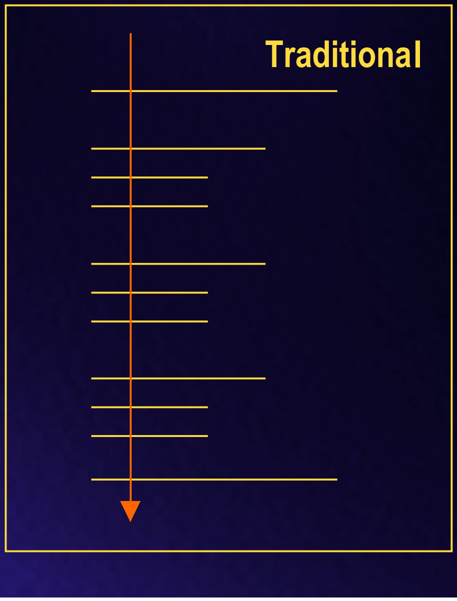
major@software-architects

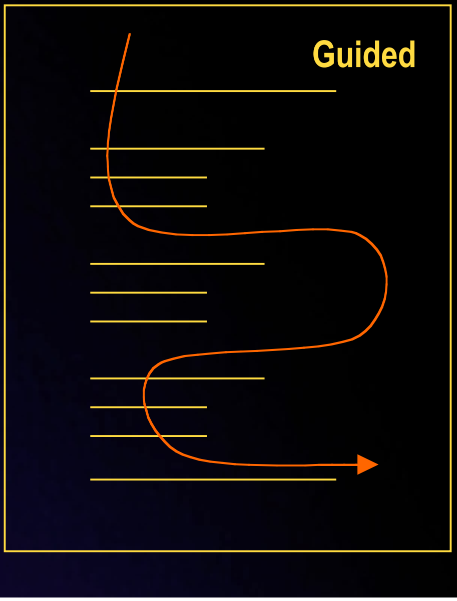
Problem

- Existing inspection/review techniques examine ***what is in the model*** for errors.
- There is no systematic way to consider ***what should be in the model***.
- ***Guided Inspection*** is a technique that supplements rigorous inspection/review techniques, that address model syntax, with test cases to systematically examine the semantics of the model.

What’s Different

- Guided Inspection does not move sequentially. What is inspected next depends upon the scenario or semantics.
- Inspection can be driven by customer priorities.
- Inspection can be focused to identify specific types of defects.





Guided Inspection Outline

- Analyze the model to be inspected.
- Complete the checklist for the appropriate model.
- Systematically sample to select test cases.
- Write down the test cases.
- Apply the test cases to the model to be inspected.
- Analyze the model to determine coverage levels.

Components in Guided Inspection

- Checklists
  - The tester completes lists by examining the products.
  - The lists are standard across products/projects.
- Test Cases
  - The tester creates test cases.
  - The developer supports a symbolic execution.
  - Tests are specific to the product.

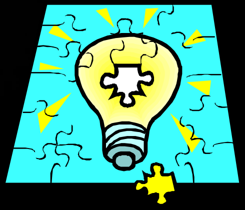
Roles in Guided Inspection

- Tester
  - Select and write test cases.

- Developer
  - Perform symbolic execution.

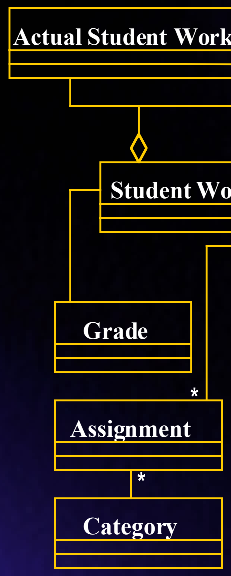
- Manager
  - Stay out of the way - this is defect finding, not a managerial evaluation.

C^3 Evaluation Criteria for Models

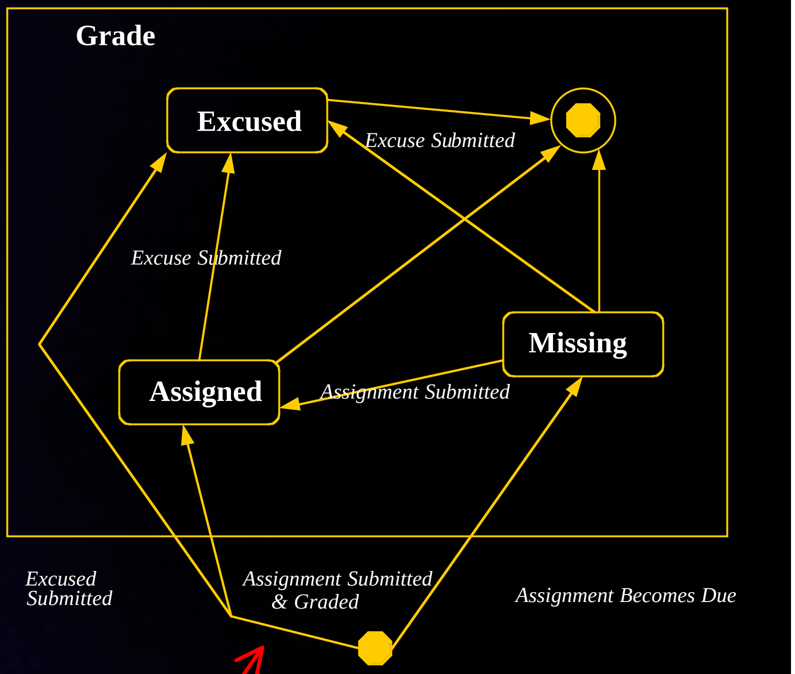


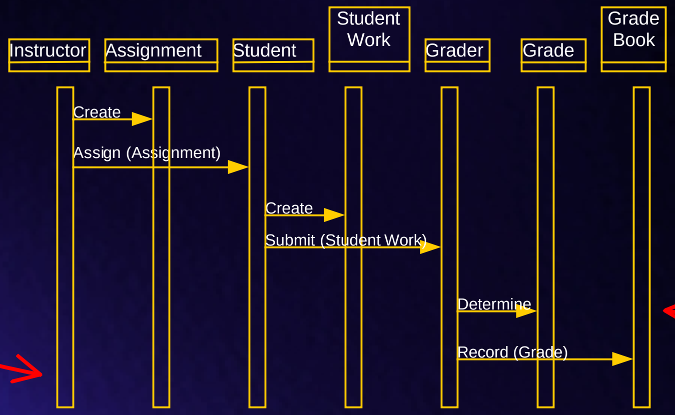
- Completeness
  - Are there scenarios the model can not handle?
  - No required elements are missing.
- Correctness
  - Does the model handle each scenario accurately?
  - Judged equivalent to a reference standard.
- Consistency
  - Are there any contradictions among elements within a work product (internal)?
  - Are there any contradictions between work products (external)?

Consistency Between Diagrams



Every trigger is a method



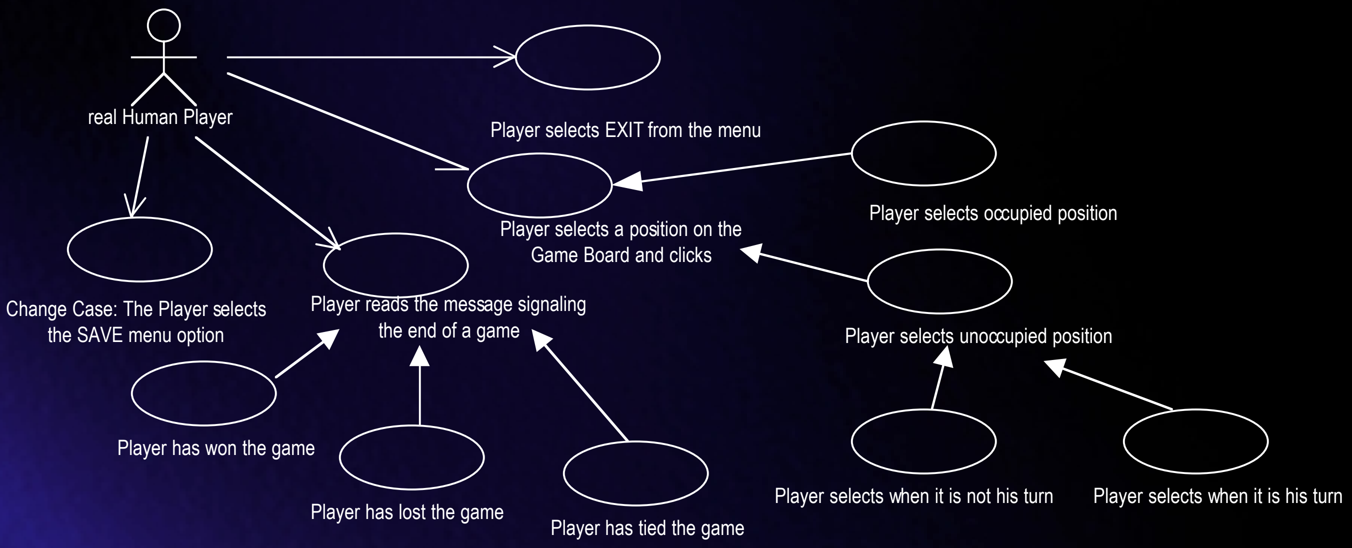


Each message is a method

Sequence of messages corresponds to state sequence

Building System Test Cases

- For analysis and high-level design models the test cases can be generated from system use cases.



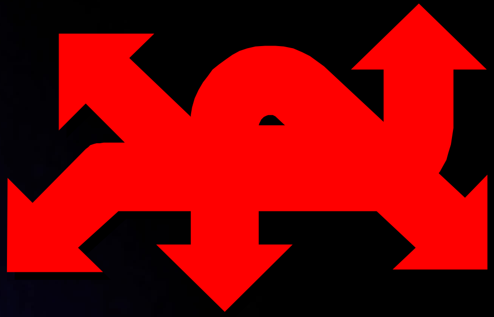
real Human Player
Player selects EXIT from the menu
Player selects occupied position
Player selects a position on the Game Board and clicks
Change Case: The Player selects the SAVE menu option
Player reads the message signaling the end of a game
Player selects unoccupied position
Player has won the game
Player has lost the game
Player has tied the game
Player selects when it is not his turn
Player selects when it is his turn

Determining Priorities

- Each use case is annotated with three attributes:
  - Frequency - How often will this feature be used relative to other features of the system?
  - Criticality - How necessary is this feature to the success of the product?
  - Risk - How likely is there to be a problem in implementing this feature?
- Each attribute is valued on a scale from **Low** to **High**.

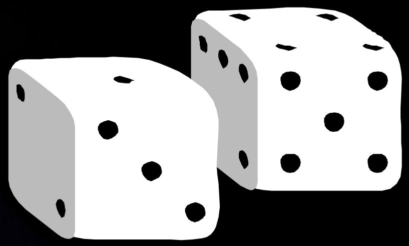
Combining Attribute Values

- For a single use case, we have three attribute values.
- Risk is used for scheduling development increments.
- Frequency and criticality are both useful for testing:
  - The most often used, most necessary feature should be tested the most.
- If criticality is **HIGH** and frequency is **LOW:**
  - **Conservative** combined value - **HIGH**
  - **Averaging** combined value - **MEDIUM**



Sampling

- Use cases that have a combined frequency/criticality rating of high will be tested over a wider range than those with a low rating.
- Equivalence classes are established for each variable.
- Test cases are formed by selecting values from the equivalence classes.
- A value for a field is chosen and paired with values of each equivalence class for each variable.

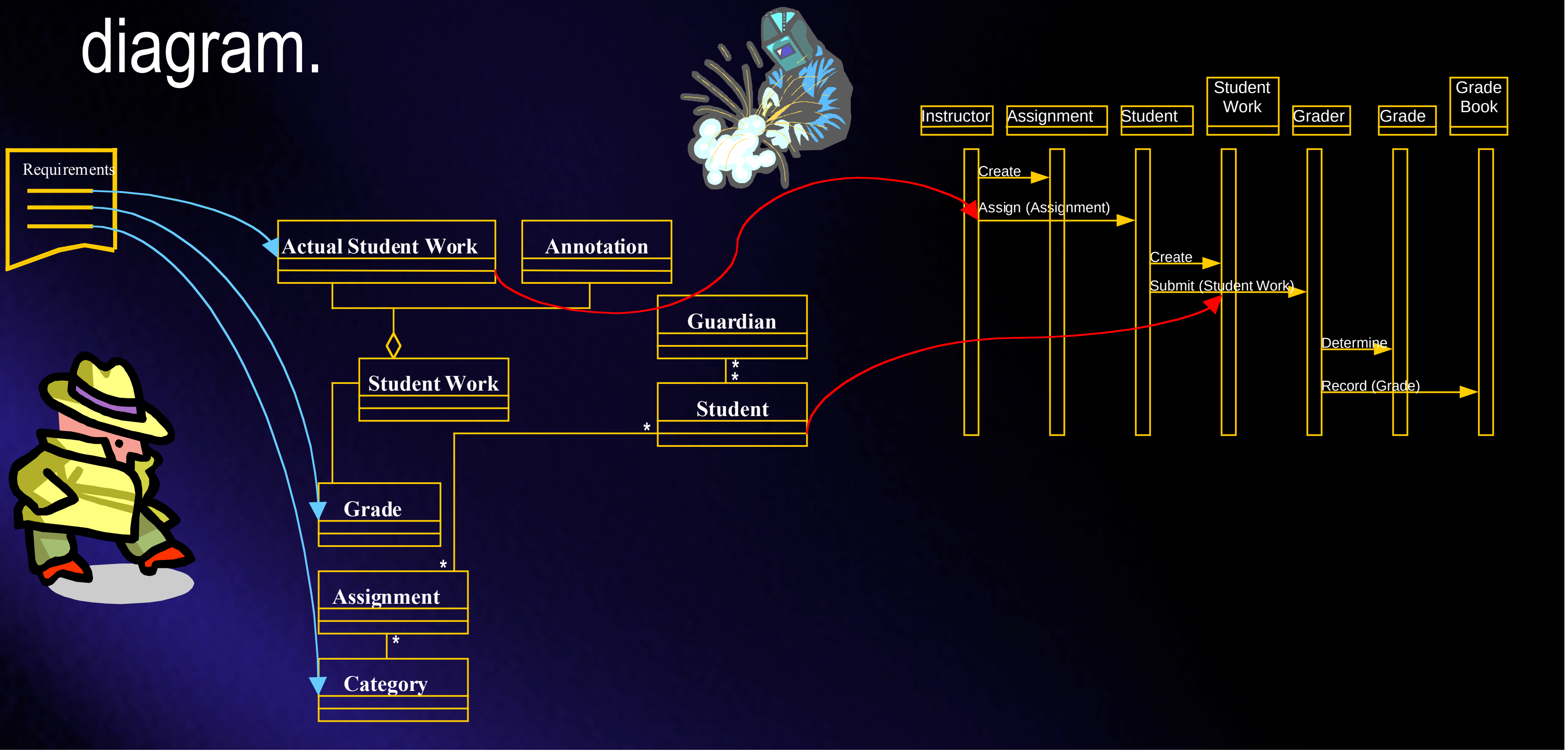


Inspection Session

- Testers guide the inspection by setting the scenario.
- Developers “describe” the execution using their knowledge of the classes, but also referring to pre and post-conditions.
- Developers record the execution using an appropriate UML diagram.

Executing a Test Case

- The scenario guides the inspection of the class diagram. The results are recorded as a sequence diagram.



Effectiveness of Guided Inspection

- Data to collect
  - Number of defects detected
  - Number of person hours
- Effectiveness
  - Yield = defects/person hour
- Analysis
  - The bigger the yield the better

Conclusion

- The system has been analyzed from three perspectives: correctness, completeness and consistency.
- Companies have reported that it costs as much as 100 times more to repair a defect at system test time as it would to repair at analysis time.
- While guided inspection is a person intensive technique, even the early expenditure of considerable resources can still result in a net savings over the full project life cycle.

Thanks

- On behalf of Software Architects, thank you for attending this session.
- A more complete presentation of this material is available on our web site:

  www.software-architects.com
- My e-mail address is:

  major@software-architects.com

Please keep in touch if there is anything I can do for you.